Quality-adaptive Prefetching for Interactive Branched Video using HTTP-based Adaptive Streaming

Vengatanathan Krishnamoorthi¹, Niklas Carlsson¹, Derek Eager², Anirban Mahanti³, Nahid Shahmehri¹

¹ Linköping university, Sweden
² University of Saskatchewan, Canada
³ NICTA, Australia

We have all seen a movie that (in our taste) is...
We have all seen a movie that (in our taste) is...

too sad
We have all seen a movie that (in our taste) is...

too sad

too violent
We have all seen a movie that (in our taste) is...

too sad
too violent
too scary

...
We have all seen a movie that (in our taste) is...

too sad

too violent

too scary

… or where we may have wanted our favorite character to make a different choice…
We have all seen a movie that (in our taste) is...

too sad

too violent

too scary

... or where we may have wanted our favorite character to make a different choice...
We have all seen a movie that (in our taste) is...

too sad

too violent

too scary

... or where we may have wanted our favorite character to make a different choice...
We have all seen a movie that (in our taste) is...

too sad
too violent
too scary

... or where we may have wanted our favorite character to make a different choice...
What if we can personalize the storyline based on the users preferences or path choices?
What if we can personalize the storyline based on the users preferences or path choices?
What if we can personalize the storyline based on the users preferences or path choices?
What if we can personalize the storyline based on the users preferences or path choices?
What if we can personalize the storyline based on the users preferences or path choices?

... already many examples how creative content creators provide interactive experiences and story lines ...
What if we can personalize the storyline based on the users preferences or path choices?

... already many examples how creative content creators provide interactive experiences and story lines ...

YouTube | Interlude | ... and even books!
What if we can personalize the storyline based on the users preferences or path choices?

… already many examples how creative content creators provide interactive experiences and story lines …

YouTube Interlude … and even books!
What if we can personalize the storyline based on the users preferences or path choices?

… already many examples how creative content creators provide interactive experiences and story lines …

YouTube

Interlude

… and even books!
What if we can personalize the storyline based on the users preferences or path choices?

... already many examples how creative content creators provide interactive experiences and story lines ...

YouTube

Interlude

... and even books!
Seamless Playback without Stalls

- Regardless of interactivity, user experience and user satisfaction is greatly influenced by:
  - Playback stalls and quality fluctuations
Seamless Playback without Stalls

• Regardless of interactivity, user experience and user satisfaction is greatly influenced by:
  – Playback stalls and quality fluctuations
Seamless Playback without Stalls

• Regardless of interactivity, user experience and user satisfaction is greatly influenced by:
  – Playback stalls and quality fluctuations
Seamless Playback without Stalls

- Regardless of interactivity, user experience and user satisfaction is greatly influenced by:
  - Playback stalls and quality fluctuations

- Current interactive branched players split a video into many sub videos and then link them

[Solution: Combine branched video and HAS (Krishnamoorthi et al., ACM CCR 2013)]
Seamless Playback without Stalls

• Regardless of interactivity, user experience and user satisfaction is greatly influenced by:
  – Playback stalls and quality fluctuations
• Current interactive branched players split a video into many sub videos and then link them
Seamless Playback without Stalls

- Regardless of interactivity, user experience and user satisfaction is greatly influenced by:
  - Playback stalls and quality fluctuations
- Current interactive branched players split a video into many sub videos and then link them
- Issues
  - Playback stalls when playing a new video
  - Non-adaptive playback
Seamless Playback without Stalls

• Regardless of interactivity, user experience and user satisfaction is greatly influenced by:
  – Playback stalls and quality fluctuations

• Current interactive branched players split a video into many sub videos and then link them

• Issues
  – Playback stalls when playing a new video
  – Non-adaptive playback
Seamless Playback without Stalls

• Regardless of interactivity, user experience and user satisfaction is greatly influenced by:
  – Playback stalls and quality fluctuations
• Current interactive branched players split a video into many sub videos and then link them
• Issues
  – Playback stalls when playing a new video
  – Non-adaptive playback
• Solution
Seamless Playback without Stalls

• Regardless of interactivity, user experience and user satisfaction is greatly influenced by:
  – Playback stalls and quality fluctuations
• Current interactive branched players split a video into many sub videos and then link them
• Issues
  – Playback stalls when playing a new video
  – Non-adaptive playback
• Solution
  – Combine branched video and HAS

[Krishnamoorthi et al., ACM CCR 2013]
Seamless Playback without Stalls

- Regardless of interactivity, user experience and user satisfaction is greatly influenced by:
  - Playback stalls and quality fluctuations
- Current interactive branched players split a video into many sub videos and then link them
- Issues
  - Playback stalls when playing a new video
  - Non-adaptive playback
- Solution
  - Combine branched video and HAS

[Krishnamoorthi et al., ACM CCR 2013]
Seamless Playback without Stalls

• Regardless of interactivity, user experience and user satisfaction is greatly influenced by:
  – Playback stalls and quality fluctuations

• Current interactive branched players split a video into many sub videos and then link them

• Issues
  – Playback stalls when playing a new video
  – Non-adaptive playback

• Solution
  – Combine branched video and HAS

[Krishnamoorthi et al., ACM CCR 2013]
Seamless Playback without Stalls

- Regardless of interactivity, user experience and user satisfaction is greatly influenced by:
  - Playback stalls and quality fluctuations
- Current interactive branched players split a video into many sub videos and then link them
- Issues
  - Playback stalls when playing a new video
  - Non-adaptive playback
- Solution
  - Combine branched video and HAS

[Krishnamoorthi et al., ACM CCR 2013]
HTTP-based Adaptive Streaming (HAS)

- HTTP-based streaming
  - Video is split into chunks
HTTP-based Adaptive Streaming (HAS)

- HTTP-based streaming
  - Video is split into chunks
HTTP-based Adaptive Streaming (HAS)

- HTTP-based streaming
  - Video is split into chunks
  - Easy firewall traversal and caching
HTTP-based Adaptive Streaming (HAS)

- HTTP-based streaming
  - Video is split into chunks
  - Easy firewall traversal and caching
  - Easy support for interactive VoD
HTTP-based Adaptive Streaming (HAS)

- HTTP-based streaming
  - Video is split into chunks
  - Easy firewall traversal and caching
  - Easy support for interactive VoD
- HTTP-based adaptive streaming
HTTP-based Adaptive Streaming (HAS)

- HTTP-based streaming
  - Video is split into chunks
  - Easy firewall traversal and caching
  - Easy support for interactive VoD

- HTTP-based adaptive streaming
  - Multiple encodings of each fragment (defined in manifest file)
HTTP-based Adaptive Streaming (HAS)

- HTTP-based streaming
  - Video is split into chunks
  - Easy firewall traversal and caching
  - Easy support for interactive VoD

- HTTP-based adaptive streaming
  - Multiple encodings of each fragment (defined in manifest file)
HTTP-based Adaptive Streaming (HAS)

- **HTTP-based streaming**
  - Video is split into chunks
  - Easy firewall traversal and caching
  - Easy support for interactive VoD

- **HTTP-based adaptive streaming**
  - Multiple encodings of each fragment (defined in manifest file)
  - Clients adapt quality encoding based on buffer/network conditions
HTTP-based Adaptive Streaming (HAS)

- **HTTP-based streaming**
  - Video is split into chunks
  - Easy firewall traversal and caching
  - Easy support for interactive VoD

- **HTTP-based adaptive streaming**
  - Multiple encodings of each fragment (defined in manifest file)
  - Clients adapt quality encoding based on buffer/network conditions
HTTP-based Adaptive Streaming (HAS)

- **HTTP-based streaming**
  - Video is split into chunks
  - Easy firewall traversal and caching
  - Easy support for interactive VoD

- **HTTP-based adaptive streaming**
  - Multiple encodings of each fragment (defined in manifest file)
  - Clients adapt quality encoding based on buffer/network conditions
HTTP-based Adaptive Streaming (HAS)

- **HTTP-based streaming**
  - Video is split into chunks
  - Easy firewall traversal and caching
  - Easy support for interactive VoD

- **HTTP-based adaptive streaming**
  - Multiple encodings of each fragment (defined in manifest file)
  - Clients adapt quality encoding based on buffer/network conditions
HTTP-based Adaptive Streaming (HAS)

- **HTTP-based streaming**
  - Video is split into chunks
  - Easy firewall traversal and caching
  - Easy support for interactive VoD

- **HTTP-based adaptive streaming**
  - Multiple encodings of each fragment (defined in manifest file)
  - Clients adapt quality encoding based on buffer/network conditions
HTTP-based Adaptive Streaming (HAS)

- **HTTP-based streaming**
  - Video is split into chunks
  - Easy firewall traversal and caching
  - Easy support for interactive VoD

- **HTTP-based adaptive streaming**
  - Multiple encodings of each fragment (defined in manifest file)
  - Clients adapt quality encoding based on buffer/network conditions
HTTP-based Adaptive Streaming (HAS)

- HTTP-based streaming
  - Video is split into chunks
  - Easy firewall traversal and caching
  - Easy support for interactive VoD

- HTTP-based adaptive streaming
  - Multiple encodings of each fragment (defined in manifest file)
  - Clients adapt quality encoding based on buffer/network conditions
HAS-based Interactive Branched Video

• Branched video and branch points
HAS-based Interactive Branched Video

- Branched video and branch points
  - The video can include branch points, with multiple branch choices
HAS-based Interactive Branched Video

- Branched video and branch points
  - The video can include branch points, with multiple branch choices
HAS-based Interactive Branched Video

• Branched video and branch points
  – The video can include branch points, with multiple branch choices
HAS-based Interactive Branched Video

• Branched video and branch points
  – The video can include branch points, with multiple branch choices
  – User selects which segment to play back next
HAS-based Interactive Branched Video

- Branched video and branch points
  - The video can include branch points, with multiple branch choices
  - User selects which segment to play back next
- Segments
HAS-based Interactive Branched Video

- Branched video and branch points
  - The video can include branch points, with multiple branch choices
  - User selects which segment to play back next
- Segments
  - Arbitrary sequence of chunks from one or more videos
HAS-based Interactive Branched Video

• Branched video and branch points
  – The video can include branch points, with multiple branch choices
  – User selects which segment to play back next

• Segments
  – **Arbitrary** sequence of chunks from one or more videos
HAS-based Interactive Branched Video

- Branched video and branch points
  - The video can include branch points, with multiple branch choices
  - User selects which segment to play back next
- Segments
  - Arbitrary sequence of chunks from one or more videos
- Use of HAS allow adaptive prefetching
HAS-based Interactive Branched Video

• Branched video and branch points
  – The video can include branch points, with multiple branch choices
  – User selects which segment to play back next

• Segments
  – Arbitrary sequence of chunks from one or more videos

• Use of HAS allow adaptive prefetching
  – Goal: Seamless playback even if user decision at last possible moment
HAS-based Interactive Branched Video

- Branched video and branch points
  - The video can include branch points, with multiple branch choices
  - User selects which segment to play back next

- Segments
  - Arbitrary sequence of chunks from one or more videos

- Use of HAS allow adaptive prefetching
  - Goal: Seamless playback even if user decision at last possible moment
Contributions

• We develop a simple analytic model which allows us to define the prefetching problem as an optimization problem
  – Maximizes expected playback quality while avoiding stalls

• Based on our findings, we design optimized policies that determine:
  1. When different chunks should be downloaded
  2. What quality level should be selected for each of these chunks
  3. How to manage playback buffers and (multiple) TCP connections such as to ensure smooth playback experience without excessive workahead (buffering)

• The design and implementation of the framework

• Experimental evaluation of our policies, which provide insights into the importance of careful adaptive policies
Contributions

• We develop a simple analytic model which allows us to define the prefetching problem as an optimization problem
  – Maximizes expected playback quality while avoiding stalls

• Based on our findings, we design optimized policies that determine:
  1. When different chunks should be downloaded
  2. What quality level should be selected for each of these chunks
  3. How to manage playback buffers and (multiple) TCP connections such as to ensure smooth playback experience without excessive workahead (buffering)

• The design and implementation of the framework

• Experimental evaluation of our policies, which provide insights into the importance of careful adaptive policies
Contributions

• We develop a simple analytic model which allows us to define the prefetching problem as an optimization problem
  — Maximizes expected playback quality while avoiding stalls

• Based on our findings, we design optimized policies that determine:
  1. When different chunks should be downloaded
  2. What quality level should be selected for each of these chunks
  3. How to manage playback buffers and (multiple) TCP connections such as to ensure smooth playback experience without excessive workahead (buffering)

• The design and implementation of the framework

• Experimental evaluation of our policies, which provide insights into the importance of careful adaptive policies
Contributions

• We develop a simple analytic model which allows us to define the prefetching problem as an optimization problem
  – Maximizes expected playback quality while avoiding stalls

• Based on our findings, we design optimized policies that determine:
  1. When different chunks should be downloaded
  2. What quality level should be selected for each of these chunks
  3. How to manage playback buffers and (multiple) TCP connections such as to ensure smooth playback experience without excessive workahead (buffering)

• The design and implementation* of the framework

• Experimental evaluation of our policies, which provide insights into the importance of careful adaptive policies

*Software: http://www.ida.liu.se/~nikca/mm14.html
Contributions

• We develop a simple analytic model which allows us to define the prefetching problem as an optimization problem
  – Maximizes expected playback quality while avoiding stalls

• Based on our findings, we design optimized policies that determine:
  1. When different chunks should be downloaded
  2. What quality level should be selected for each of these chunks
  3. How to manage playback buffers and (multiple) TCP connections such as to ensure smooth playback experience without excessive workahead (buffering)

• The design and implementation* of the framework

• Experimental evaluation of our policies, which provide insights into the importance of careful adaptive policies

*Software: http://www.ida.liu.se/~nikca/mm14.html
Contributions

• We develop a simple analytic model which allows us to define the prefetching problem as an optimization problem
  – Maximizes expected playback quality while avoiding stalls

• Based on our findings, we design optimized policies that determine:
  1. When different chunks should be downloaded
  2. What quality level should be selected for each of these chunks
  3. How to manage playback buffers and (multiple) TCP connections such as to ensure smooth playback experience without excessive workahead (buffering)

• The design and implementation* of the framework

• Experimental evaluation of our policies, which provide insights into the importance of careful adaptive policies

*Software: http://www.ida.liu.se/~nikca/mm14.html
Contributions

• We develop a simple analytic model which allows us to define the prefetching problem as an optimization problem
  – Maximizes expected playback quality while avoiding stalls

• Based on our findings, we design optimized policies that determine:
  1. When different chunks should be downloaded
  2. What quality level should be selected for each of these chunks
  3. How to manage playback buffers and (multiple) TCP connections such as to ensure smooth playback experience without excessive workahead (buffering)

• The design and implementation* of the framework

• Experimental evaluation of our policies, which provide insights into the importance of careful adaptive policies

*Software: http://www.ida.liu.se/~nikca/mm14.html
Problem Description and Constraints

• Problem: Maximize quality, given playback deadlines and bandwidth conditions
Problem Description and Constraints

- Problem: Maximize quality, given playback deadlines and bandwidth conditions
Problem Description and Constraints

• Objective function

\[
\text{maximize } \text{playback quality}
\]
Problem Description and Constraints

- Objective function

\[
\text{maximize } \sum_{i=1}^{n_e} q_i l_i + \sum_{i=n_e+1}^{n_e+|E^b|} q_i l_i
\]
Problem Description and Constraints

• Objective function

\[
\text{maximize } \sum_{i=1}^{n_e} q_i l_i + \sum_{i=n_e+1}^{n_e+|\mathcal{E}^b|} q_i l_i
\]
Problem Description and Constraints

• Objective function

\[
\text{maximize} \sum_{i=1}^{n_e} q_i l_i + \sum_{i=n_e+1}^{n_e+|E^b|} q_i l_i
\]
Problem Description and Constraints

• Download order: round robin (optimal)
Problem Description and Constraints

• Download order: round robin (optimal)
Problem Description and Constraints

- Download order: round robin (optimal)
Problem Description and Constraints

• Download order: round robin (optimal)
Problem Description and Constraints

• Download order: round robin (optimal)
Problem Description and Constraints

- Download order: round robin (optimal)
Problem Description and Constraints

- Download order: round robin (optimal)
Problem Description and Constraints

- Download order: round robin (optimal)
Problem Description and Constraints

• Download order: round robin **(extra workahead)**
Problem Description and Constraints

- Download order: round robin (extra workahead)

Current segment  first chunk next  extra workahead
Problem Description and Constraints

- Download order: round robin (extra workahead)
Problem Description and Constraints

- Download order: round robin (extra workahead)
Problem Description and Constraints

- Once branch point has been traversed, move on to next segment ...

```
<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
</table>
```

Current segment  

```
| 1 | 2 | 3 | 4 | 7 | 10 |
```

first chunk next
Problem Description and Constraints

- Once branch point has been traversed, move on to next segment ...
Problem Description and Constraints

• Once branch point has been traversed, move on to next segment ...

Selected path

current segment
Problem Description and Constraints

- Once branch point has been traversed, move on to next segment ...
Problem Description and Constraints

- Once branch point has been traversed, move on to next segment ...
Problem Description and Constraints

• Once branch point has been traversed, move on to next segment ...
Problem Description and Constraints

- Once branch point has been traversed, move on to next segment ...
Problem Description and Constraints

• Once branch point has been traversed, move on to next segment ...
Problem Description and Constraints

- Playback deadlines
  - for seamless playback without stalls
Problem Description and Constraints

- Playback deadlines
  - for seamless playback without stalls

Playback schedule

Download schedule

\[
\begin{array}{cccccc}
1 & 2 & 3 & 4 & 7 & 10 \\
\end{array}
\]
Problem Description and Constraints

- Playback deadlines
  - for seamless playback without stalls
  - Current segment: e.g., 2 and 3
Problem Description and Constraints

• Playback deadlines
  – for seamless playback without stalls
  – Current segment: e.g., 2 and 3

\[ t_i^c \leq t_i^d = \tau + \sum_{j=1}^{i-1} l_j, \quad \text{if } 1 \leq i \leq n_e \]
Problem Description and Constraints

- Playback deadlines
  - for seamless playback without stalls
  - Current segment: e.g., 2 and 3

\[ t_i^c \leq t_i^d = \tau + \sum_{j=1}^{i-1} l_j \text{, if } 1 \leq i \leq n_e \]
Problem Description and Constraints

- Playback deadlines
  - for seamless playback without stalls
  - Current segment: e.g., 2 and 3

\[ t_i^c \leq t_i^d = \tau + \sum_{j=1}^{i-1} l_j, \]  \text{ if } 1 \leq i \leq n_e

Download completion time

Download completion times
Problem Description and Constraints

- Playback deadlines
  - for seamless playback without stalls
  - Current segment: e.g., 2 and 3

\[ t_i^c \leq t_i^d = \tau + \sum_{j=1}^{i-1} l_j, \quad \text{if } 1 \leq i \leq n_e \]
Problem Description and Constraints

• Playback deadlines
  – for seamless playback without stalls
  – Current segment: e.g., 2 and 3

\[ t^c_i \leq t^d_i = \tau + \sum_{j=1}^{i-1} l_j, \quad \text{if } 1 \leq i \leq n_e \]
Problem Description and Constraints

• Playback deadlines
  – for seamless playback without stalls
  – Current segment: e.g., 2 and 3

\[ t_i^c \leq t_i^d = \tau + \sum_{j=1}^{i-1} l_j, \quad \text{if } 1 \leq i \leq n_e \]
Problem Description and Constraints

• Playback deadlines
  – for seamless playback without stalls
  – Current segment: e.g., 2 and 3

\[ t_{i}^{c} \leq t_{i}^{d} = \tau + \sum_{j=1}^{i-1} l_{j}, \quad \text{if } 1 \leq i \leq n_{e} \]
Problem Description and Constraints

• Playback deadlines
  – for seamless playback without stalls
  – First chunks next segment: e.g., 4, 7, and 10
Problem Description and Constraints

• Playback deadlines
  – for seamless playback without stalls
  – First chunks next segment: e.g., 4, 7, and 10

\[ t_i^c \leq t_i^d = \tau + \sum_{j=1}^{n_e} l_j, \] if \( n_e < i \leq n_e + |E^b| \)
Problem Description and Constraints

- Playback deadlines
  - for seamless playback without stalls
  - First chunks next segment: e.g., 4, 7, and 10

\[ t_i^c \leq t_i^d = \tau + \sum_{j=1}^{n_e} l_j, \quad \text{if } n_e < i \leq n_e + |E^b| \]
Problem Description and Constraints

• Playback deadlines
  – for seamless playback without stalls
  – First chunks next segment: e.g., 4, 7, and 10

\[
t_i^c \leq t_i^d = \tau + \sum_{j=1}^{n_e} l_j, \quad \text{if } n_e < i \leq n_e + |\mathcal{E}^b|
\]

Time at which branch point is reached

Capture completion times
Problem Description and Constraints

\[ t_i^c \leq t_i^d = \tau + \sum_{j=1}^{n_e} l_j, \quad \text{if } n_e < i \leq n_e + |E^b| \]

Download completion times
Problem Description and Constraints

- Download times $t_{i}^{c}$, rate estimations, and parallel connections
Problem Description and Constraints

- Download times $t_i^c$, rate estimations, and parallel connections
  - At the end of a chunk download, schedule new downloads and new TCP connections
  - Assume that an additional TCP connection will not increase the total download rate
  - New connections are initiated only if it is not expected to lead to playback deadline violations
Problem Description and Constraints

• Download times $t^c_i$, rate estimations, and parallel connections
  – At the end of a chunk download, schedule new downloads and new TCP connections
  – Assume that an additional TCP connection will not increase the total download rate
  – New connections are initiated only if it is not expected to lead to playback deadline violations
Problem Description and Constraints

• Download times $t^c_i$, rate estimations, and parallel connections
  – At the end of a chunk download, schedule new downloads and new TCP connections
  – Assume that an additional TCP connection will not increase the total download rate
  – New connections are initiated only if it is not expected to lead to playback deadline violations
Problem Description and Constraints

• Download times $t_i^c$, rate estimations, and parallel connections
  – At the end of a chunk download, schedule new downloads and new TCP connections
  – Assume that an additional TCP connection will not increase the total download rate
  – New connections are initiated only if it is not expected to lead to playback deadline violations
Concurrent Download Example
Concurrent Download Example
Concurrent Download Example
Concurrent Download Example
Concurrent Download Example
Concurrent Download Example
Concurrent Download Example
Prefetching Policies

• At download completion
  – Decide number of chunks to download next
  – Decide quality level of chunks
  – Maximize expected weighted playback
Prefetching Policies

• At download completion
  – Decide number of chunks to download next
  – Decide quality level of chunks
  – Maximize expected weighted playback
Prefetching Policies

• At download completion
  – Decide number of chunks to download next
  – Decide quality level of chunks
  – Maximize expected weighted playback
Prefetching Policies

• At download completion
  – Decide number of chunks to download next
  – Decide quality level of chunks
  – Maximize expected weighted playback
Prefetching Policies

• At download completion
  – Decide number of chunks to download next
  – Decide quality level of chunks
  – Maximize expected weighted playback

• Exponential number of candidate schedules
Prefetching Policies

• At download completion
  – Decide number of chunks to download next
  – Decide quality level of chunks
  – Maximize expected weighted playback
• Exponential number of candidate schedules
• Our optimized policies restrict the number of candidate schedules to consider
  – Policies differ in number of candidate schedules and how aggressive they are (in choosing qualities)
## Comparison Between Policies

<table>
<thead>
<tr>
<th>Policy</th>
<th>Connections</th>
<th>Schedules considered</th>
<th>Objective</th>
</tr>
</thead>
<tbody>
<tr>
<td>All schedules</td>
<td>$1 \leq c_i \leq C^{\text{max}}$</td>
<td>$Q^M$, where $M=n_e+</td>
<td>\xi_b</td>
</tr>
<tr>
<td>Optimized non-increasing quality</td>
<td>$1 \leq c_i \leq C^{\text{max}}$</td>
<td>$\begin{cases} M+Q-1 \ Q-1 \end{cases}$</td>
<td>$\sum_{i=1}^{n_e} q_i/l_i + \sum_{i=n_e+1}^{n_e+</td>
</tr>
<tr>
<td>Optimized maintainable quality</td>
<td>$1 \leq c_i \leq C^{\text{max}}$</td>
<td>$Q$</td>
<td></td>
</tr>
</tbody>
</table>

- **Total number of schedules**: $Q^M$
- **Optimized non-increasing quality**:  
  - Constraint: Qualities of consecutive chunks are non-increasing
- **Optimized maintainable quality**:  
  - Constraint: Chosen quality must be sustainable for the remaining chunks
## Comparison Between Policies

<table>
<thead>
<tr>
<th>Policy</th>
<th>Connections</th>
<th>Schedules considered</th>
<th>Objective</th>
</tr>
</thead>
<tbody>
<tr>
<td>All schedules</td>
<td>$1 \leq c_i \leq C_{\text{max}}$</td>
<td>$Q^M$, where $M = n_e +</td>
<td>\xi_b</td>
</tr>
<tr>
<td>Optimized non-increasing quality</td>
<td>$1 \leq c_i \leq C_{\text{max}}$</td>
<td>$\sum_{i=1}^{n_e} q_i l_i + \sum_{i=n_e+1}^{n_e+</td>
<td>\xi_b</td>
</tr>
<tr>
<td>Optimized maintainable quality</td>
<td>$1 \leq c_i \leq C_{\text{max}}$</td>
<td>$Q$</td>
<td>$Q$</td>
</tr>
</tbody>
</table>

- Total number of schedules: $Q^M$
- **Optimized non-increasing quality:**
  - Constraint: Qualities of consecutive chunks are non-increasing
- **Optimized maintainable quality:**
  - Constraint: Chosen quality must be sustainable for the remaining chunks
# Comparison Between Policies

<table>
<thead>
<tr>
<th>Policy</th>
<th>Connections</th>
<th>Schedules considered</th>
<th>Objective</th>
</tr>
</thead>
<tbody>
<tr>
<td>All schedules</td>
<td>1≤c_i≤C^{max}</td>
<td>Q^M, where M=n_e +</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>\mid \xi_b \mid -m</td>
<td></td>
</tr>
<tr>
<td>Optimized non-</td>
<td>1≤c_i≤C^{max}</td>
<td>\begin{pmatrix} M+Q-1 \ Q-1 \end{pmatrix}</td>
<td>\sum_{i=1}^{n_e} q_i/<em>{i} + \sum</em>{i=n_e+1}^{n_e+</td>
</tr>
<tr>
<td>increasing quality</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Optimized maintainable quality</td>
<td>1≤c_i≤C^{max}</td>
<td>Q</td>
<td></td>
</tr>
</tbody>
</table>

- **Total number of schedules:** Q^M
- **Optimized non-increasing quality:**
  - Constraint: Qualities of consecutive chunks are non-increasing
- **Optimized maintainable quality:**
  - Constraint: Chosen quality must be sustainable for the remaining chunks
## Comparison Between Policies

<table>
<thead>
<tr>
<th>Policy</th>
<th>Connections</th>
<th>Schedules considered</th>
<th>Objective</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single connection</td>
<td>1</td>
<td>Q</td>
<td>( \sum_{i=1}^{n_e} q_i l_i + \sum_{i=n_e+1}^{n_e+</td>
</tr>
<tr>
<td>Greedy bandwidth</td>
<td>1 ≤ c_i ≤ C_{\text{max}}</td>
<td>Q</td>
<td>( \sum_{i=j}^{j+m} q_i l_i )</td>
</tr>
</tbody>
</table>

- **Single connection**: baseline comparing to policies which do not use multiple connections
- **Greedy bandwidth**: bandwidth aggressive as opposed to aggressive quality choices
- **Naïve**: benchmark to regular branched video players
## Comparison Between Policies

<table>
<thead>
<tr>
<th>Policy</th>
<th>Connections</th>
<th>Schedules considered</th>
<th>Objective</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single connection</td>
<td>1</td>
<td></td>
<td>[ \sum_{i=1}^{n_e} q_i l_i + \sum_{i=n_e+1}^{n_e+</td>
</tr>
<tr>
<td>Greedy bandwidth</td>
<td>1≤c_i≤C_{\text{max}}</td>
<td></td>
<td>[ \frac{j+m}{\sum_{i=j}^{n_e} l_i} ]</td>
</tr>
</tbody>
</table>

- Single connection: baseline comparing to policies which do not use multiple connections
- Greedy bandwidth: bandwidth aggressive as opposed to aggressive quality choices
- Naïve: benchmark to regular branched video players
## Comparison Between Policies

<table>
<thead>
<tr>
<th>Policy</th>
<th>Connections</th>
<th>Schedules considered</th>
<th>Objective</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single connection</td>
<td>1</td>
<td></td>
<td>$\sum_{i=1}^{n_e} q_i l_i + \sum_{i=n_e+1}^{n_e+</td>
</tr>
<tr>
<td>Greedy bandwidth</td>
<td>$1 \leq c_i \leq C^{\text{max}}$</td>
<td></td>
<td>$\sum_{i=j}^{j+m} q_i l_i$</td>
</tr>
</tbody>
</table>

- Single connection: baseline comparing to policies which do not use multiple connections
- Greedy bandwidth: bandwidth aggressive as opposed to aggressive quality choices
- Naïve: benchmark to regular branched video players
Test Scenario
Test Scenario

Worst case scenario
• always pick the last segment
• at last possible moment
Test Scenario

Worst case scenario
- always pick the last segment
- at last possible moment
Test Scenario

Default scenario:
- Chunks per segment: 5
- Branches per branch point: 4
- Branch points: 3

Worst case scenario:
- always pick the last segment
- at last possible moment
**Test Scenario**

- Default scenario:
  - Chunks per segment: 5
  - Branches per branch point: 4
  - Branch points: 3

Results are averages over 30 experiments.
Test Scenario

• Default scenario:
  – Chunks per segment: 5
  – Branches per branch point: 4
  – Branch points: 3
Test Scenario

• Default scenario:
  – Chunks per segment: 5
  – Branches per branch point: 4
  – Branch points: 3
Test Scenario

- Default scenario:
  - Chunks per segment: 5
  - Branches per branch point: 4
  - Branch points: 3
Test Scenario

- Default scenario:
  - Chunks per segment: 5
  - Branches per branch point: 4
  - Branch points: 3

- Results are averages over 30 experiments
Policy Comparison

- Naïve policy: does not perform prefetching
  - Stalls at every branch point
  - Note: High playback rate is misleading on its own
• Naïve policy: does not perform prefetching
  – Stalls at every branch point
  – Note: High playback rate is misleading on its own
• Naïve policy: does not perform prefetching
  – Stalls at every branch point
  – Note: High playback rate is misleading on its own
Policy Comparison

- *Optimized maintainable quality* provides best tradeoff
Policy Comparison

- **Optimized maintainable quality** provides best tradeoff
  - Much lower stall probability
  - Tradeoff is somewhat lower playback rate
Policy Comparison

- **Optimized maintainable quality** provides best tradeoff
  - Much lower stall probability
  - Tradeoff is somewhat lower playback rate
Policy Comparison

- **Optimized maintainable quality** provides best tradeoff
  - Much lower stall probability
  - Tradeoff is somewhat lower playback rate
Policy Comparison

- **Optimized non-increasing quality** is more aggressive
  - Higher playback rate
  - More stalls
Policy Comparison

- **Optimized non-increasing quality** is more aggressive
  - Higher playback rate
  - More stalls
Policy Comparison

- **Optimized non-increasing quality** is more aggressive
  - Higher playback rate
  - More stalls
• *Single connection* does not use parallel connections
  – Good (slightly higher) playback rate
  – Much more stalls
Policy Comparison

- **Single connection** does not use parallel connections
  - Good (slightly higher) playback rate
  - Much more stalls
• **Single connection** does not use parallel connections
  – Good (slightly higher) playback rate
  – Much more stalls
Policy Comparison

- **Greedy bandwidth** aggressively grabs bandwidth
  - Lower playback rate
  - More stalls
Policy Comparison

- **Greedy bandwidth** aggressively grabs bandwidth
  - Lower playback rate
  - More stalls
• **Greedy bandwidth** aggressively grabs bandwidth
  – Lower playback rate
  – More stalls
Impact of Round-trip Times (RTTs)

- Quality decreases with larger RTTs
  - Playback rate decrease with RTT
  - Stall probability increase with RTT
Impact of Round-trip Times (RTTs)

- Quality decreases with larger RTTs
  - Playback rate decrease with RTT
  - Stall probability increase with RTT
Impact of Round-trip Times (RTTs)

- Quality decreases with larger RTTs
  - Playback rate decrease with RTT
  - Stall probability increase with RTT
Impact of Segment Lengths
Impact of Segment Lengths

- Quality increases with more chunks per segment
- Very many stalls if segments are too short
Impact of Segment Lengths

- Quality increases with more chunks per segment
- Very many stalls if segments are too short
Impact of Segment Lengths

- Quality increases with more chunks per segment
- Very many stalls if segments are too short
Impact of Branch Options
Impact of Branch Options

- Stalls frequent when too many branch options
  - Single connection struggles the most
Impact of Branch Options

- Stalls frequent when too many branch options
  - Single connection struggles the most
Impact of Branch Options

- Stalls frequent when too many branch options
  - Single connection struggles the most
Impact of Competing Flows
Impact of Competing Flows

- Player adapts playback rate based on competing traffic
- Parallel connection polices see increased benefits when competing traffic
  - E.g., Single connection policy has much more stalls when competing flows
Impact of Competing Flows

- Player adapts playback rate based on competing traffic
- Parallel connection polices see increased benefits when competing traffic
  - E.g., Single connection policy has much more stalls when competing flows
Impact of Competing Flows

- Player adapts playback rate based on competing traffic
- Parallel connection polices see increased benefits when competing traffic
  - E.g., Single connection policy has much more stalls when competing flows
Impact of Competing Flows

- Player adapts playback rate based on competing traffic
- Parallel connection polices see increased benefits when competing traffic
  - E.g., Single connection policy has much more stalls when competing flows
**Impact of Competing Flows**

- Player adapts playback rate based on competing traffic
- Parallel connection polices see increased benefits when competing traffic
  - E.g., Single connection policy has much more stalls when competing flows
Capped Workahead

• Most HAS players perform ON-OFF switching based on two buffer thresholds: $T_{\text{min}}$ and $T_{\text{max}}$
  • If buffer $> T_{\text{min}}$
    – Start playback
  • If buffer $> T_{\text{max}}$
    – Suspend download
  • If buffer $< T_{\text{min}}$
    – Resume download
Most HAS players perform ON-OFF switching based on two buffer thresholds: $T_{\text{min}}$ and $T_{\text{max}}$

- If buffer > $T_{\text{min}}$ – Start playback
- If buffer > $T_{\text{max}}$ – Suspend download
- If buffer < $T_{\text{min}}$ – Resume download
Capped Workahead

- Most HAS players perform ON-OFF switching based on two buffer thresholds: $T_{min}$ and $T_{max}$
  - If buffer > $T_{min}$ – Start playback
  - If buffer > $T_{max}$ – Suspend download
  - If buffer < $T_{min}$ – Resume download
Capped Workahead

- Most HAS players perform ON-OFF switching based on two buffer thresholds: $T_{\text{min}}$ and $T_{\text{max}}$
  - If buffer $> T_{\text{min}}$ → Start playback
  - If buffer $> T_{\text{max}}$ → Suspend download
  - If buffer $< T_{\text{min}}$ → Resume download
Most HAS players perform ON-OFF switching based on two buffer thresholds: $T_{\text{min}}$ and $T_{\text{max}}$.

- If buffer $> T_{\text{min}}$ – Start playback
- If buffer $> T_{\text{max}}$ – Suspend download
- If buffer $< T_{\text{min}}$ – Resume download
Capped Workahead

- Most HAS players perform ON-OFF switching based on two buffer thresholds: $T_{min}$ and $T_{max}$
  - If buffer $> T_{min}$
    - Start playback
  - If buffer $> T_{max}$
    - Suspend download
  - If buffer $< T_{min}$
    - Resume download
Capped Workahead

• Most HAS players perform ON-OFF switching based on two buffer thresholds: $T_{\text{min}}$ and $T_{\text{max}}$
  • If buffer $> T_{\text{min}}$  
    – Start playback
  • If buffer $> T_{\text{max}}$  
    – Suspend download
  • If buffer $< T_{\text{min}}$  
    – Resume download
Capped Workahead

- Most HAS players perform ON-OFF switching based on two buffer thresholds: $T_{min}$ and $T_{max}$
- If buffer > $T_{min}$
  - Start playback
- If buffer > $T_{max}$
  - Suspend download
- If buffer < $T_{min}$
  - Resume download
Capped Workahead

- Most HAS players perform ON-OFF switching based on two buffer thresholds: $T_{\text{min}}$ and $T_{\text{max}}$
  - If buffer > $T_{\text{min}}$  
    - Start playback
  - If buffer > $T_{\text{max}}$  
    - Suspend download
  - If buffer < $T_{\text{min}}$  
    - Resume download
Most HAS players perform ON-OFF switching based on two buffer thresholds: $T_{\text{min}}$ and $T_{\text{max}}$

- If buffer > $T_{\text{min}}$  
  – Start playback
- If buffer > $T_{\text{max}}$  
  – Suspend download
- If buffer < $T_{\text{min}}$  
  – Resume download
Capped Workahead

• How to handle workahead when video contains branches?

\[ T_{\text{min}} = T_{\text{single}} \times (\# \text{branches}) \]

\[ T_{\text{max}} = T_{\text{single}} + \Delta \]
Capped Workahead

• How to handle workahead when video contains branches?

• Perform ON-OFF switching based on number of branches after the closest branch point
  
  • $T_{\text{min}} = T_{\text{single}} \cdot (\# \text{ branches})$
  
  • $T_{\text{max}} = T_{\text{single}} + \Delta$

![Diagram of branching pathways](image)
Capped Workahead

• How to handle workahead when video contains branches?

• Perform ON-OFF switching based on number of branches after the closest branch point
  • \( T_{\text{min}} = T_{\text{single}} \cdot (\# \text{ branches}) \)
  • \( T_{\text{max}} = T_{\text{single}} + \Delta \)
Capped Workahead

• How to handle workahead when video contains branches?
• Perform ON-OFF switching based on number of branches after the closest branch point
  • $T_{\text{min}} = T_{\text{single}} \cdot (# \text{ branches})$
  • $T_{\text{max}} = T_{\text{single}} + \Delta$

Example
Capped Workahead

• How to handle workahead when video contains branches?

• Perform ON-OFF switching based on number of branches after the closest branch point
  
  • $T_{\text{min}} = T_{\text{single}} \cdot (# \text{ branches})$
  
  • $T_{\text{max}} = T_{\text{single}} + \Delta$

Example

$$T_{\text{single}} = 8$$
$$\Delta = 4$$
# branches = 2
Capped Workahead

• How to handle workahead when video contains branches?

• Perform ON-OFF switching based on number of branches after the closest branch point

  • $T_{\text{min}} = T_{\text{single}} \cdot (# \text{ branches})$
  • $T_{\text{max}} = T_{\text{single}} + \Delta$

Example

```
T_{\text{single}} = 8
\Delta = 4
# \text{ branches} = 2

\{ T_{\text{min}} = 16 \}
\{ T_{\text{max}} = 20 \}
```
Capped Workahead

• How to handle workahead when video contains branches?
• Perform ON-OFF switching based on number of branches after the closest branch point
  - \( T_{\text{min}} = T_{\text{single}} \cdot (# \text{ branches}) \)
  - \( T_{\text{max}} = T_{\text{single}} + \Delta \)

Example
Conclusion

• Designed and implemented branched video player that achieve seamless streaming without playback interruptions

• Designed optimized policies that maximize playback quality while ensuring sufficient workahead to avoid stalls

• Evaluation shows that solution effectively adapt quality levels and number of parallel connections so as to provide best possible video quality, given current conditions
Conclusion

• Designed and implemented branched video player that achieve seamless streaming without playback interruptions

• Designed optimized policies that maximize playback quality while ensuring sufficient workahead to avoid stalls

• Evaluation shows that solution effectively adapt quality levels and number of parallel connections so as to provide best possible video quality, given current conditions
Conclusion

• Designed and implemented branched video player that achieve seamless streaming without playback interruptions

• Designed optimized policies that maximize playback quality while ensuring sufficient workahead to avoid stalls

• Evaluation shows that solution effectively adapt quality levels and number of parallel connections so as to provide best possible video quality, given current conditions
Conclusion

• Designed and implemented branched video player that achieve seamless streaming without playback interruptions

• Designed optimized policies that maximize playback quality while ensuring sufficient workahead to avoid stalls

• Evaluation shows that solution effectively adapt quality levels and number of parallel connections so as to provide best possible video quality, given current conditions
Conclusion

• Designed and implemented branched video player that achieve seamless streaming without playback interruptions
• Designed optimized policies that maximize playback quality while ensuring sufficient workahead to avoid stalls
• Evaluation shows that solution effectively adapt quality levels and number of parallel connections so as to provide best possible video quality, given current conditions

Software: http://www.ida.liu.se/~nikca/mm14.html
Quality-adaptive Prefetching for Interactive Branched Video using HTTP-based Adaptive Streaming

Vengatanathan Krishnamoorthi, Niklas Carlsson, Derek Eager, Anirban Mahanti, Nahid Shahmehri

Software: http://www.ida.liu.se/~nikca/mm14.html